JOURNAL OF COMPLEXITY 5, 107–117 (1989)

A Monte Carlo Method for the Parallel Solution of Linear Systems

BRUNO CODENOTTI

Istituto di Elaborazione dell' Informazione del CNR, Vis S. Maria 46, 56100 Pisa, Italy

AND

FRANCO FLANDOLI

Dipartmento di Mathematica, Universita' di Torino, Via Principe Amedeo 8, 10123 Torino, Italy

Received January 6, 1987

A new parallel algorithm for the solution of linear systems, based upon the Monte Carlo approach, is shown. The method allows one to obtain the solution of a linear system with parallel cost growing as the logarithm of the size of the coefficient matrix, and with "probabilistic" error bounded in terms of the Chebyshev inequality. © 1989 Academic Press, Inc.

1. INTRODUCTION

In this paper we show new methods, based on the Monte Carlo approach, for solving linear systems which can be efficiently implemented in a parallel computational environment.

Throughout the paper, we shall assume that the following parallel model is used:

(i) any number of processors can be used at any time;

(ii) each processor may perform one of the arithmetic or logical operations;

(iii) each operation takes one unit of time (one *time step*);

0885-064X/89 $3.00

(iv) no time is required to communicate data among processors;

(v) there are no memory or data alignment penalties.

It is worth reporting that such assumptions are popular in the current literature.

Recently several authors have investigated the problem of devising efficient parallel algorithms for the solution of linear systems, as well as for matrix inversion (see, for example, Csanky, 1976; Pan and Reif, 1985). The time bound attained by the above-mentioned methods is $O(\log^2 n)$, where n is the size of the matrix. It turns out that there is still a gap between the trivial logarithmic lower bound and the upper bounds.

In (Pan and Reif, 1985) the time cost $O(\log^2 n)$ is obtained by assuming that the condition number of the coefficient matrix is upper bounded by a power of n and is under the conditions of applicability of Newton's method.

In this paper, we give a first look at the application of Monte Carlo methods (see Shreider, 1966, for the features of Monte Carlo methods) to the parallel solution of linear systems (as well as to matrix inversion). We need assumptions analogous to those used in (Pan and Reif, 1985) to attain the time bound $O(\log n)$ by a probabilistic algorithm (see the proposition of Section 4).

In the following,

$\|A\|$ denotes the norm of the matrix A, induced by the infinite vector norm $\|\cdot\|$ (the use of different norms will be explicitly reported),

$\rho(A)$ denotes the spectral radius of the matrix A, $\langle \cdot, \cdot \rangle$ denotes a scalar product between vectors, $\operatorname{sign}(z)$ denotes the sign of a real number z, $\lceil z \rceil$ denotes the upper integer part of a real number z, $E(X)$ denotes the mean value of a random variable X, $V(X)$ denotes the variance of a random variable X, $\operatorname{Cov}(X)$ denotes the covariance of a random variable X.

The paper is organized as follows: In Section 2, one basic Monte Carlo algorithm based on the notion of Markov chains is illustrated, which is essentially taken from (Forsythe and Leibler, 1950). Section 3 concerns the parallel computation derived by the Monte Carlo algorithm presented in Section 2. Section 4 analyzes the time cost, as well as the number of processors, of the parallel solution of linear systems, with respect to the size of the problem. In Section 5, we extend the results of the previous sections.

2. Basic Monte Carlo Methods

Let $B = (b_{ij})$ be an $n \times n$ matrix with real entries, and $q = (q_i)$ be a real n-vector. In this section, we show a Monte Carlo algorithm for the computation of one entry of $\sum_{i=0}^{R} B^i q$, for fixed r.

Let

$$x^R = \sum_{i=0}^{R} B^i q,$$

and, componentwise,

$$x_m^R = \sum_{r \le R} \sum_{i_1 i_2 \ldots i_r} b_{m i_1} b_{i_1 i_2} \cdot \cdot \cdot b_{i_{r-1} i_r} q_{i_r}. \tag{2.1}$$

Assume that

$$b_{ij} = f_{ij} p_{ij}, \qquad 0 \le p_{ij} \le 1, \qquad i, j = 1, 2, \ldots, n,$$

$$q_i = g_i p_i, \qquad 0 \le p_i \le 1, \qquad \sum_{j=1}^{n} p_{ij} + p_i = 1, \qquad i = 1, 2, \ldots, n.$$

Equality (2.1) can now be rewritten as

$$x_m^R = \sum_{r \le R} \sum_{i_1 i_2 \ldots i_r} f_{m i_1} f_{i_1 i_2} \cdot \cdot \cdot f_{i_{r-1} i_r} g_{i_r} p_{m i_1} p_{i_1 i_2} \cdot \cdot \cdot p_{i_{r-1} i_r}. \tag{2.2}$$

We are now able to describe a stochastic method for the evaluation of x_m^R by means of (2.2). Let $X_0, X_1, \ldots, X_R$ be a finite Markov chain with states $1, 2, \ldots, n+1$, transition probabilities

$$P(X_{i+1} = l | X_i = k) = \begin{cases} p_{kl} & \text{if } k \le n \text{ and } l \le n, \\ p_k & \text{if } l = n+1 \text{ and } k \le n, \\ 1 & \text{if } l = k = n+1, \\ 0 & \text{if } k = n+1, \text{ and } l \le n, \end{cases}$$

and initial distribution concentrated in m, i.e., $P(X_0 = m) = 1$. The structure of the above-described Markov chain is illustrated in Fig. 2.1.

The $(R+1)$-uple $m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1, i_j = n+1, j = 1, \ldots, r$, denotes a trajectory of the Markov chain, and $P(m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1)$ denotes the probability of the trajectory. From the Markov property (see Pan and Reif, 1985), we have

$$P(m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1) = \begin{cases} p_{m i_1} p_{i_1 i_2} \cdot \cdot \cdot p_{i_{r-1} i_r} p_{i_r}, & r < R, \\ p_{m i_1} p_{i_1 i_2} \cdot \cdot \cdot p_{i_{R-1} i_R}, & r = R. \end{cases} \tag{2.3}$$

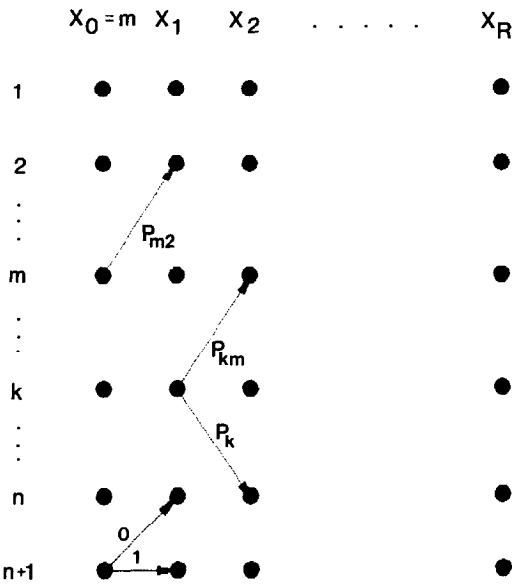

FIG. 2.1. The Markov chain.

Let Ω be the set of all trajectories. Let $\mathbf{F}$ be the σ-algebra of all subsets of Ω, and let P be the probability on $(\Omega, \mathbf{F})$ defined by (2.3). Let ϕ_m^R: $\Omega \to \mathbb{R}$ be the random variable defined as

$$\phi_m^R(m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1) = \begin{cases} f_{mi_1} f_{i_1 i_2} \cdots f_{i_{r-1} i_r} g_{i_r} & \text{if } r < R, \\ f_{mi_1} f_{i_1 i_2} \cdots f_{i_{R-1} i_R} q_{i_R} & \text{if } r = R. \end{cases} \tag{2.4}$$

It is easy to see that $E(\phi_m^R) = x_m^R$. Indeed, by definition of expectation, along with (2.3) and (2.4), we have

$$\begin{aligned} E(\phi_m^R) &= \sum_{r \le R} \sum_{i_1 i_2 \ldots i_r} P(m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1) \\ &\quad \phi_m^R(m, i_1, i_2, \ldots, i_r, n+1, \ldots, n+1) \\ &\quad + \sum_{i_1 i_2 \ldots i_R} P(m, i_1, i_2, \ldots, i_R) \phi_m^R(m, i_1, i_2, \ldots, i_R) \\ &= \sum_{r \le R} \sum_{i_1 i_2 \ldots i_r} p_{mi_1} p_{i_1 i_2} \cdots p_{i_{r-1} i_r} p_{i_r} f_{mi_1} f_{i_1 i_2} \cdots f_{i_{r-1} i_r} g_{i_r} \\ &\quad + \sum_{i_1 i_2 \ldots i_R} p_{mi_1} p_{i_1 i_2} \cdots p_{i_{R-1} i_R} f_{mi_1} f_{i_1 i_2} \cdots f_{i_{R-1} i_R} q_{i_R} = x_m^R, \end{aligned}$$

by (2.2).

The arguments discussed above suggest an algorithm—described in Section 3—for the approximated computation of x_m^R.

3. A Parallel Algorithm for the Solution of Linear Systems

In this section we present a parallel algorithm for the solution of linear systems, based on the Monte Carlo method described in Section 2, under the assumption $\|B\| < 1$. Such an assumption will be relaxed in Section 5, where we will discuss the case $\|B\| = 1$.

In the following, we shall choose

$$p_{ij} = |b_{ij}|, \qquad f_{ij} = \operatorname{sign}(b_{ij}), \qquad i, j = 1, \ldots, n$$

and, consequently,

$$p_i = 1 - \sum_{j=1}^{n} p_{ij}, \qquad g_i = q_i/p_i, \qquad i = 1, \ldots, n.$$

Note that the assumption $\|B\| < 1$ implies $p_i \neq 0$, $i = 1, \ldots, n$. Moreover, it follows from (2.4) that

$$|\phi_m^R| \leq \|q\|/(1 - \|B\|),$$

so that

$$V(\phi_m^R) = E((\phi_m^R)^2) - (E(\phi_m^R))^2 \leq \|q\|^2/(1 - \|B\|)^2.$$

We now present the parallel algorithm.

Algorithm

Stage 1. Compute $P\,(p_{ij})$, $F = (f_{ij})$, $i, j = 1, 2, \ldots, n$;
Stage 2. Compute the matrix $Q = (q_{ij})$, where

$$q_{ij} = \sum_{k=1}^{j} p_{ik}, \; i, j = 1, 2, \ldots, n;$$

Stage 3. Compute $p_i = 1 - q_{in}$, $i = 1, \ldots, n$;
Stage 4. Compute $g_i = q_i/p_i$, $i = 1, \ldots, n$;
Stage 5. Compute N trajectories $w_1, \ldots, w_N$:
for each w_i:

5.1. Produce, independently, numbers x_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, R$, randomly chosen in [0, 1];

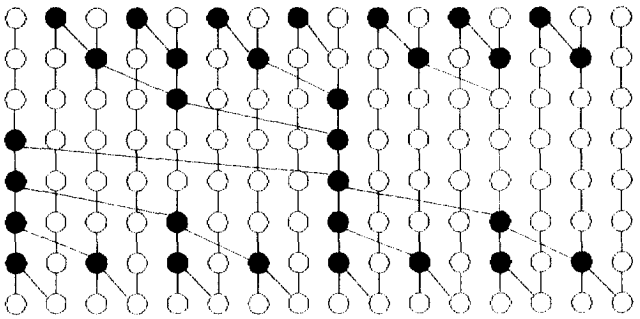

FIG. 3.1. Computation graph of the parallel algorithm for the computation of all the partial sums $q_{ij}, j = 1, \ldots, n$, on n processors [case $n = 16$].

5.2. Determine integers h_{ij} for which $q_{ih_{ij}} \leq x_{ij} \leq q_{ih_{ij}+1}$, $i, j = 1, \ldots, n$, i.e., a matrix $H = (h_{ij})$, and consequently, a trajectory $j_0, j_1, \ldots, j_n$, where

$$j_0 = m;$$

$$j_i = L_{j_{i-1},i}.$$

Stage 6. Compute $\phi_m^R(w_i)$, $i = 1, \ldots, N$;
Stage 7. Compute $x_m^R = (\sum_{i=1}^N \phi_m^R(w_i))/N$.

We turn now to the evaluation of the parallel cost of the algorithm, together with the number of processors sufficient to perform the computation. Stage 1 can be carried out in a straightforward way, with unitary time cost on $2n^2$ processors. Stage 2 can be performed in $2\lceil \log n \rceil$ time steps on n^2 processors, by using the well-known algorithm (Brent and Kung, 1981) described in Fig. 3.1. Stages 3 and 4 can be carried out in one time step each, on n processors. Stage 5 computes in parallel N trajectories. Each trajectory can be computed in time $\lceil \log n \rceil$ on nR processors, since step 5.1 requires constant time on nR processors, and step 5.2 can be performed by using a binary search algorithm with running time $\lceil \log n \rceil$ on nR processors, and then the indexes j_i are available in matrix H. Therefore the overall number of processors is nNR. Stage 6 consists of the product of at most R factors, so that $\lceil \log R \rceil$ time steps suffice on NR processors. Finally, Stage 7 can be performed in time $\lceil \log N \rceil + 1$ on $N - 1$ on processors. This results in a global time cost of the algorithm of

$$2\lceil \log n \rceil + \lceil \log R \rceil + \lceil \log N \rceil + O(1),$$

on $\max\{2n^2, nNR\}$ processors.

In the next section, we will compare the quantities N, R, and the size n

of the problem, and we will attain the time bound $O(\log n)$ for the Monte Carlo algorithm described above.

4. Parallel Cost of a Monte Carlo Linear Systems Solver

In this section, we will evaluate the cost of computing an approximated solution $\tilde{x}$ of the linear system $Ax = q$, satisfying

$$\|x - \tilde{x}\|/\|x\| \leq eps, \tag{4.1}$$

where *eps* is a positive constant, "a priori" fixed.

In the following, we will make the choice eps $= 2^{-d+1}$, where d is the number of arithmetic digits, i.e., *eps* equal to the computer relative precision. Let A be an $n \times n$ matrix, and assume

$$A = I - B, \qquad \text{where } \|B\| < 1. \tag{4.2}$$

Note that a matrix A of the form (4.2) can be obtained by transformations of more general matrices, as occurs, for example, when constructing iterative methods. Moreover let q be an n-vector. The linear system $Ax = q$ can be rewritten as

$$x = Bx + q. \tag{4.3}$$

In this section, we derive the parallel cost of applying the algorithm of the previous section to the solution of (4.3). It is now worth recalling the Chebyshev inequality (Shreider, 1966). Let Y be a random variable, with $E(Y) = \mu$, and $V(Y) = \sigma^2$. Let Y_i, $i = 1, \ldots, N$, be N independent random variables with the same distribution as Y. Let $\mu_N = (\sum_{i=1}^{N} Y_i)/N$. We have

$$P\{|\mu_N - \mu| > \lambda\} < \sigma^2/\lambda^2 N, \qquad \lambda > 0. \tag{4.4}$$

If we apply inequality (4.4) and relation (3.1) to evaluate the behavior of the algorithm, we obtain

$$P\left\{\left|x_m^R - \left(\sum_{i=1}^{N} \phi_m^R(\omega_i)\right)\Big/N\right| > \lambda\right\} < \max_{1\leq i\leq n} q_i^2/\lambda^2 N(1 - \|B\|)^2. \tag{4.5}$$

Let $\tilde{x} = \sum_{i=1}^{N} \phi_m^R(\omega_i)/N$. We have

$$\|x - \tilde{x}\|/\|x\| \leq \|\tilde{x} - x^R\|/\|x\| + \|x^R - x\|/\|x\|.$$

Then we will impose

$$\|\tilde{x} - x^R\|/\|x\| \leq 2^{-d} \tag{4.6}$$

$$\|x^R - x\|/\|x\| \leq 2^{-d}. \tag{4.7}$$

Since $\|x^R - x\| \leq \|B\|^{R+1}\|x\|$, then R can be chosen according to $\|B\|^{R+1} \leq 2^{-d}$, i.e., $R \geq d/\log 1/\|B\| - 1$. If we now assume $\|B\| \leq 1 - 1/\theta(n^k)$, then we can choose $R = \Omega(dn^k)$ in order to satisfy (4.7).

Note that (4.6) can only be satisfied in terms of the Chebychev inequality. Given $\delta > 0$, we look for $N > 1$ such that

$$P(\|\tilde{x} - x^R\|/\|x\| > 2^{-d}) < \delta,$$

where $\tilde{x} = \sum_{i=1}^{N} \phi_m^R(\omega_i)/N$. We have

$$P(\|\tilde{x} - x^R\|/\|x\| > 2^{-d}) = P\left(\bigcup_{m=1}^{n} (|\tilde{x}_m - x_m^R|/\|x\| > 2^{-d})\right)$$

$$\leq \sum_{m=1}^{n} P(|\tilde{x}_m - x_m^R|/\|x\| > 2^{-d}).$$

Therefore it is sufficient to choose N such that

$$P(|\tilde{x}_m - x_m^R|/\|x\| > 2^{-d}) < \delta/n, \qquad \text{for any } m.$$

From (4.5) with $\lambda = 2^{-d}\|x\|$, we have

$$2^{2d}\|q\|^2/N(1 - \|B\|)^2\|x\|^2 < \delta/n,$$

i.e.,

$$N > 2^{2d}\|q\|^2/\delta(1 - \|B\|)^2\|x\|^2.$$

Note that $\|q\|/\|x\| \leq 2$, since $\|q\| \leq \|A\|\,\|x\|$, and $\|A\| \leq 2$. Therefore one can choose

$$N = \Omega(2^{2d}\delta^{-1}n^{2k+1}),$$

recalling the assumption on $\|B\|$.

The above-described inequalities and the results of the previous sections allow one to derive the following proposition.

PROPOSITION. *Let B be an $n \times n$ matrix satisfying $\|B\| \leq 1 - 1/\theta(n^k)$, and let q be an n-vector. Given a positive constant δ, an approximate solution $\tilde{x}$ of the linear system $x = Bx + q$ can be obtained in time $O(\log n)$ on $O(n^{3+3k})$ processors by the Monte Carlo method, corresponding to the algorithm of Section 3, for which $P\{\|x - \tilde{x}\|/\|x\| \leq 2^{-d}\} > 1 - \delta$, where 2^{-d+1} is the machine precision.*

COROLLARY. *If $\|B\| \leq \lambda < 1$, then $O(n^3)$ processors are sufficient to compute in $O(\log n)$ the approximation solution $\tilde{x}$, as in the previous proposition.*

5. EXTENSIONS AND FURTHER RESULTS

In this section, we present two different methods to handle the case $\|B\| = 1$.

The first method is based on the following lemma.

LEMMA. *Let x^t be the solution of the linear system*

$$x^t = tBx^t + q,$$

for each $0 < t < 1$. Given $\varepsilon > 0$, the inequality

$$\|x^t - x\|/\|x\| < \varepsilon$$

holds if

$$t > 1 - \varepsilon/(1 + \varepsilon)\|A^{-1}\|.$$

Proof. Recall the resolvent identity

$$(I - tB)^{-1} - (I - B)^{-1} = (I - tB)^{-1}(t - 1)B(I - B)^{-1}$$

It follows that

$$x^t - x = (I - tB)^{-1}(t - 1)Bx \tag{5.1}$$

and also that

$$\|(I - tB)^{-1}\| \leq \|(I - B)^{-1}\| + (1 - t)\|(I - tB)^{-1}\| \, \|B\| \, \|(I - B)^{-1}\|$$

from which

$$\|(I - tB)^{-1}\| \leq \|A^{-1}\|/(1 - (1 - t)\|A^{-1}\|). \tag{5.2}$$

From (5.1) and (5.2), we have

$$\|x^t - x\|/\|x\| \leq -1 + 1/(1 - (1 - t)\|A^{-1}\|).$$

The conclusion follows by a standard manipulation. ■

Thus, if we assume that $\|A^{-1}\| = O(n^k)$, it is possible to choose a value of $t < 1$ such that

(i) $\|x^t - x\|/\|x\| \leq 2^{-d}$;

(ii) the matrix tB satisfies the assumptions of the proposition of Section 4, and, in particular, $\|tB\| \leq 1 - 1/\theta(n^k)$. (5.3)

It turns out that it is sufficient to compute an approximation of x^t using the method of the previous sections, and the proposition of Section 4 holds in this case, too.

The second method consists of an algorithm for the computation of all powers $(B^h q)_m$ separately. It works under slightly different assumptions, namely,

$$\|B\| = 1, \qquad r(B) \leq 1 - 1/0(n^k), \qquad \text{and} \qquad B \text{ symmetric.}$$

In this case, the value of R can be obtained in terms of the spectral matrix norm, instead of the usual $\|\cdot\|$, and the asymtotic bounds are equivalent to the ones obtained in the rest of the paper.

In order to evaluate one power $(B^h q)_m$, we define the following stochastic process.

Assume that $b_{ij} = f_{ij}p_{ij}$, $0 \leq p_{ij} \leq 1$, $i, j = 1, 2, \ldots, n$, $\sum_{j=1}^{n} p_{ij} = 1$, $i = 1, 2, \ldots, n$. Let $X_0, X_1, \ldots, X_h$ be a finite Markov chain with states 1, 2, . . . , n, transition probabilities $P(X_{i+1} = 1 | X_i = k) = p_{kl}$, and initial distribution concentrated in m. The $(h + 1)$-uple $m, i_1, i_2, \ldots, i_h$ denotes a trajectory of the Markov chain, and $P(m, i_1, i_2, \ldots, i_h) = p_{mi_1}p_{i_1i_2} \cdots p_{i_{h-1}i_h}$ is its probability. On the space Ω of all trajectories, we define the random variable μ_m^h: $\Omega \to \mathbb{R}$ as

$$\mu_m^h(m, i_1, \ldots, i_h) = f_{mi_1}f_{i_1i_2} \cdots f_{i_{h-1}i_h}q_{i_h}.$$

It is easy to see that $E(\mu_m^h) = (B^h q)_m$. This leads to an algorithm analogous to the one described in Sections 3 and 4.

Finally, a more general algorithm well suited for parallel implementation, as well as the previous ones, can be derived by the following observations.

Let X be an n-dimensional random variable with density

$$g(X) = ce^{-1/2\langle AX,X\rangle},$$

where $A = A^T$ is positive definite, and $c = \sqrt{\det(A)/(2\pi)^n}$. We have

$$\mathrm{Cov}(X) = A^{-1},$$

and

$$(A^{-1})_{ij} = \int_{\mathbb{R}^n} x_i x_j c e^{-1/2\langle AX,X\rangle} dx$$

$$\approx \left(\sum_{k=1}^{N} x_i x_j c e^{-1/2\langle AX^k,X^k\rangle}\right) \Big/ \sum_{k=1}^{M} e^{-1/2\langle AX^k,X^k\rangle},$$

for randomly chosen x_i^k in a sufficiently large region of $\mathbb{R}^n$.

A similar algorithm is described in (Shreider, 1966).

Such an algorithm applies to positive definite matrices. This is not a loss of generality, since, given a nonsingular matrix B, we have

$$B^{-1} = (B^TB)^{-1}B^T,$$

and therefore B^{-1} can be computed in time $O(\log n)$, starting from the inverse of the positive definite matrix B^TB.

References

Brent, R. P., and Kung, H. T. (1981), The area–time complexity of binary multiplication, *J. Assoc. Comput. Mach.* **28**, 3, 521–534.

Csanky, C. (1976), Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **4**, 618–623.

Forsythe, G. E., and Leibler, R. A. (1950), Matrix inversion by a Monte Carlo method, *MTAC* **4**, 127–129.

Pan, V., and Reif, J. (1985), Efficient parallel solution of linear systems, *in* "Proceedings of the 17th ACM Symp. on the Theory of Comput., May 1985."

Shreider, Y. A. (Ed.) (1966), "The Monte Carlo Method," Pergamon, Elmsford, NY.